MetaAutomation
A Pattern Language to Bridge from Automation to Team Actions for Quality

Matt Griscom matt@wisewillow.org

MetaAutomation
A Pattern Language of software quality for faster, more scalable, and more actionable measurements and communications automatically created and directed to the team members that need to know.
MetaAutomation

Why patterns?
Why a pattern language?

MetaAutomation

“Meta” addresses the high-level values of automation, or why you would do automation in the first place, in order to be mindful and make those values better.
MetaAutomation

Making Automation:
• Faster
• Better Scale
• More Trustworthy
• More Actionable
• +Business Continuity
• Cheaper

MetaAutomation

In Scope:
• Regression automation
• Perf testing
• TIP
• TDD

Not in Scope:
• Unit testing
• Security
• Model-based Analytics
MetaAutomation

Target Audience:

- SDETs
- Test Managers
- Decision Makers
- Product Owners
- Test Leads
- Program Managers
- Dev Leads and Managers
- Quality

Software Product (SUT)
Automation

Software Product (SUT)

Software Product (SUT)
MetaAutomation, sequence and network
Checks and Tests

• Checks are a kind of test
• Any repeatable automated test is a check

Why use “Check?”

• Common pattern of creating software automation process starts with manual tests
• When the manual test becomes automated, the quality value is very different
• “Check” reduces risk by alerting the team to missing value of automated tests, so avoids the “automation fallacy”
• “Check” makes room for different best practices
Atomic Check

• All dependencies are present (E2E)
Atomic Check

• All dependencies are present (E2E)
• No Setup or Teardown handlers

Atomic Check

• All dependencies are present (E2E)
• No Setup or Teardown handlers
• Checks are atomic
Atomic Check

• All dependencies are present (E2E)
• No Setup or Teardown handlers
• Checks are atomic
• Checks can be run in any set, any order

Atomic Check

• All dependencies are present (E2E)
• No Setup or Teardown handlers
• Checks are atomic
• Checks can be run in any set, any order
• Verifications: preliminary vs. target, implicit vs. explicit
Atomic Check

- All dependencies are present (E2E)
- No Setup or Teardown handlers
- Checks are atomic
- Checks can be run in any set, any order
- Verifications: preliminary vs. target, implicit vs. explicit
- Logs vs. Artifacts

Fail soon, fail fast
Atomic Check

- All dependencies are present (E2E)
- No Setup or Teardown handlers
- Checks are atomic
- Checks can be run in any set, any order
- Verifications: preliminary vs. target, implicit vs. explicit
- Logs vs. Artifacts
- Fail soon, fail fast
- Failures are actionable
MetaAutomation
Automation at any time, including before checkin

- Self-documenting check code
  - Testers do analysis on **business logic** through check artifacts

Telemetry
Passively driving the product (at dogfooding time or with live product usage)

- Instrumented software product
  - Testers do DDQ / analysis on **aggregate data** through telemetry

Even if your product is fully instrumented, you still need automation!

Contact Information

**Matt Griscom**
matt@wisewillow.org

... and
MetaAutomation blog, twitter, LinkedIn, etc.

The book on MetaAutomation will be available by the end of 2014, first in hard copy, later on electronic media.